FUNDAMENTAL PRINCIPLES OF SOFTWARE ENGINEERING – A JOURNEY

Pierre Bourque[1], Robert Dupuis, Alain Abran, James W. Moore[2], Leonard Tripp[3], Sybille Wolff

Abstract

A set of fundamental principles can act as an enabler in the establishment of a discipline; however, software engineering still lacks a universally recognized set of fundamental principles. This article presents a progress report on an attempt to identify and develop a consensus on a set of candidate fundamental principles. A fundamental principle is less specific and more enduring than methodologies and techniques. It should be phrased to withstand the test of time. It should not contradict a more general engineering principle and should have some correspondence with "best practice." It should be precise enough to be capable of support and contradiction and should not conceal a tradeoff. It should also relate to one or more computer science or engineering concepts The proposed candidate set of fundamental principles were identified through two workshops, two Delphi studies and a web-based survey.

1. INTRODUCTION

In the IEEE collection of standards, software engineering is defined as:

> *"(1) The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, i.e. the application of engineering to software. (2) The study of approaches as in (1)."* (IEEE 610.12)

It is not easy to find, isolate and articulate the relevant principles that are fundamental to a discipline. Even in the more mature disciplines, where the fundamental principles are putatively known, knowledge is often tacit. In the emerging discipline of software engineering, some attempts have been made, e.g. (Boehm, 1983; Davis, 1995), but a consensus has not yet developed.

In the 50 year history of software, various methodologies, methods and techniques have been proposed to facilitate the development of software responsive to needs. Most have proved to be more specific to the then-current state of technology than was understood at the time. As a result, most have proved to be less universally applicable than originally intended. Despite a plethora of conferences and workshops over the past decades, and numerous periodicals, books and courses in the field, software engineering continues to lack universally recognized fundamental principles (McConnell, 1997; 1999).

---

[1] Pierre Bourque, Robert Dupuis, Alain Abran and Sybille Wolff are with the Software Engineering Management Research Laboratory at the Computer Science Department, Université du Québec à Montréal (Québec) Canada. Phone: (514) 987-3000, ext. 0315, 3479, 8900 and 0376, fax: (514) 987-8477; e-mail: bourque.pierre@uqam.ca, dupuis.robert@uqam.ca, abran.alain@uqam.ca, sybille.wolff@lrgl.uqam.ca.

[2] James W. Moore, Standards Coordinator, WC3 Center, The Mitre Corporation, 1820 Dolley Madison Blvd., McLean, Virginia, USA, 22102-3481. Phone: (703) 883-7396; fax: (703) 883-5432; e-mail: james.w.moore@ieee.org.

[3] Leonard Tripp is 1999 President of the IEEE Computer Society and Immediate Past-Chair of the IEEE Software Engineering Standards Committee. Leonard Tripp is a technical fellow in software engineering at the Boeing Company in Seattle USA. Phone: +1 206-662-4437; fax: +1 206-662-4437; email: l.tripp@computer.org

Our interest in the identification of the fundamental principles of software engineering results from work in the development of software engineering practice standards. It is widely posited that practice standards should be based upon observation, recording and consensual validation of implemented "best practices." This strategy has resulted, though, in the development of a corpus of standards that are sometimes alleged to be isolated, unconnected and *dis*-integrated, because each standard performs a local optimization of a single observed practice. It is hoped that the identification of a set of fundamental principles will provide a broad and rich framework for establishing relationships among groups of practice standards. A set of fundamental principles of the field could also help characterize the activities that differentiate software engineering from other computer-related activities and could help better define training programs. The identification of principles viewed as fundamental by the software engineering community would also provide a rich framework for analyzing and improving the Guide to the Software Engineering Body of Knowledge (Bourque, 1999). This guide aims to provide a topical access to the core subset of knowledge that characterizes the software engineering discipline.

This paper presents a progress report on work carried out to identify and develop a consensus on a set of fundamental software engineering principles[4]. The paper begins by discussing what are the criteria for recognizing fundamental principles, what are their roles and how they relate to underlying concepts. The overview of research methodology section presents each project phase: two workshops, two Delphi studies and a web-based survey. The deliverables section presents the set of candidate fundamental principles, the degree of consensus on this candidate set, and the participants who took part in the project phases. Finally, a summary of the paper, and steps that could be undertaken to improve the set of candidate principles are presented in the last section.

2. WHAT ARE FUNDAMENTAL PRINCIPLES?

Underlying Concepts versus Fundamental Principles

In discussing fundamental principles, there is sometimes confusion between such principles and what may be characterized as "underlying concepts." Table 1 contrasts the two:

Underlying Concepts	Fundamental Principles
Scientific	Engineering
Descriptive	Prescriptive
Validated through experiment	Validated through rigorous (but not necessarily experimental) assessment of practice
Judged on the basis of correctness	Judged on the basis of usability, relevance, significance, usefulness

Table 1: Underlying Concepts versus Fundamental Principles

Underlying concepts are to be regarded as scientific statements. They must be capable of validation by experiment and are judged on the basis of their correctness when subjected to experiment. By contrast, fundamental principles are to be regarded as engineering statements which prescribe constraints on solutions to problems or constraints on the process of developing solutions. They should be rigorously evaluated, but in practice rather than in the laboratory, and judged by whether or not they provide useful and substantial contributions to the successful solution of real problems of significant size and scope. In general, we would expect fundamental engineering principles to be strongly related to underlying scientific concepts.

[4] For more detailed information on this research, please see http://www.lrgl.uqam.ca/fpse

Roles of Fundamental Principles

Figure 1 illustrates the relationships sought among principles, standards and practices. It is believed that a body of fundamental principles for engineering and some other disciplines already exists and has been articulated. (Most of the relevant disciplines have a history far longer than software engineering.) Software engineering principles would, in the general case, be regarded as specializations of the principles. The software engineering principles would play the role of organizing, motivating, explaining and validating the practice standards. Implemented practices should be based on those practice standards.

Working from the specific toward the general, practice standards would be recordings and idealizations of observed and validated "best" practices. The software engineering principles would be abstractions of the practice standards. Furthermore, software engineering principles might be candidates for generalization to the status of general engineering principles, particularly when complexity is a concern.

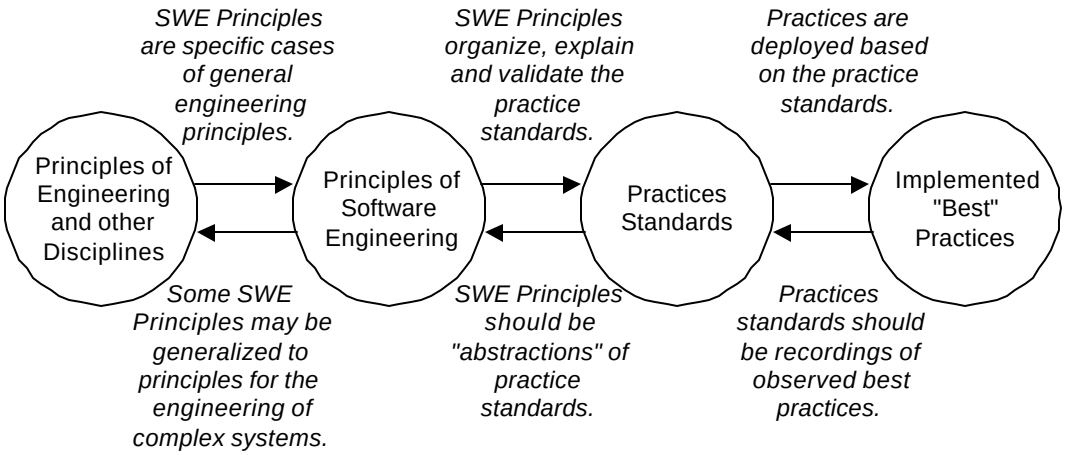

Figure 1. Relationship between principles and practice

Criteria for the recognition of fundamental principles

The project has developed the following criteria (possibly better regarded as heuristics or meta-principles) for the recognition of fundamental principles:

- Fundamental principles are less specific than methodologies and techniques, i.e. specific methodologies and techniques may be selected, *within a particular technological context*, to accomplish the intent of fundamental principles.
- Fundamental principles are more enduring than methodologies and techniques, i.e. fundamental principles should be phrased in a way that will stand the "test of time" rather than in the context of current technology.
- Fundamental principles are typically discovered or abstracted from practice and should have some correspondence with "best" practice.
- Software engineering fundamental principles should not contradict more general fundamental principles …
- … but, there may be tradeoffs in the application of fundamental principles.
- A fundamental principle should not conceal a tradeoff. By that we mean that a fundamental principle should not attempt to prioritize or select from among various qualities of a solution; the engineering process should do that. Fundamental principles should identify or explain the importance of the various qualities among which the engineering process will make trades.
- A fundamental principle should be precise enough to be capable of support or contradiction.
- A fundamental principle should relate to one or more underlying concepts.

3. OVERVIEW OF RESEARCH METHODOLOGY

The project was prompted by a 1996 decision of the IEEE Software Engineering Standards Executive Committee to begin efforts to identify a list of fundamental principles for software engineering. The research methodology that has been followed to identify a candidate list is summarized in Figure 2 and included the following steps:

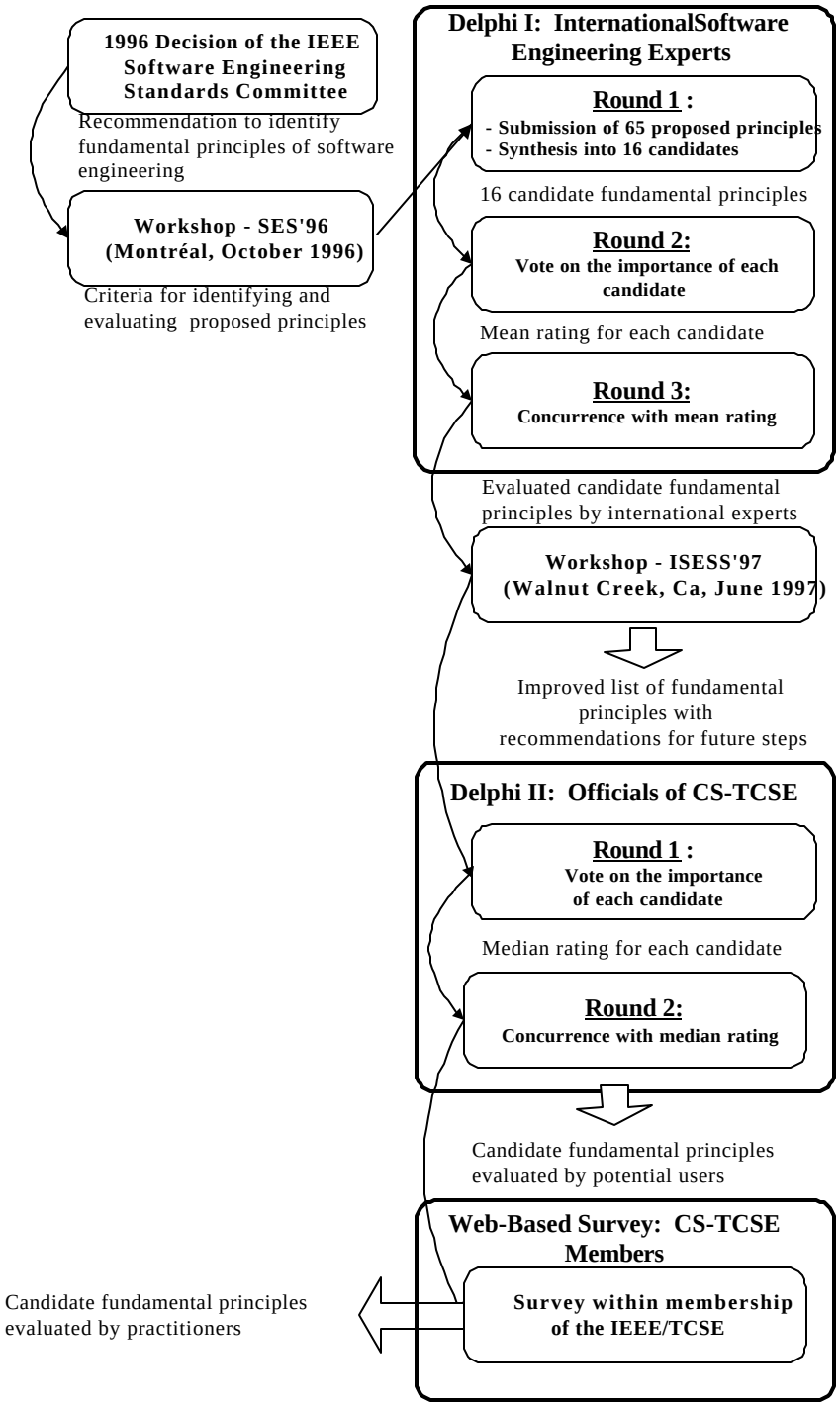

Figure 2. Overview of Project Steps

- First IEEE workshop: A workshop was organized in Montreal in October 1996 at the IEEE Software Engineering Standards Symposium (SES96) to discuss the issue of fundamental principles, to lay out the ground work and to propose an approach for identifying them. This workshop also developed a list of characteristics of fundamental principles, the process for identifying them and the criteria for searching. The Delphi approach was recommended for identifying an initial set of candidate fundamental principles. A Delphi study seeks to create a consensus among a group around a controversial question. Furthermore, the Delphi approach requires the anonymity of contributors until the end of the study, thus eliminating the influence of past relationships among participants.
- First Delphi study: A group of international experts was asked to propose a set of candidate fundamental principles. The Delphi methodology was selected to produce this initial list and to determine the degree of consensus among the international experts.
- Second IEEE workshop: Based on the output and lessons learned of the first Delphi study, a second workshop was organized in 1997 to review results and to prepare an improved list of candidate fundamental principles.
- Second Delphi study: This improved list was the input for a second Delphi study, this time with a larger group of potential users of the candidate fundamental principles. In fact, they were software engineering officials of the IEEE Computer Society. The degree of consensus was recorded for further study.
- Web-based survey: To verify the extent of the consensus among software engineering practitioners on this set of candidate fundamental principles, a web-based survey was designed and circulated to over 3000 members of the Software Engineering Technical Council[5] of the IEEE Computer Society. Again, the degree of consensus among the respondents was recorded and could be compared to the previous groups opinions.

This paper therefore presents the opinion-based process designed to obtain a set of candidate fundamental principles and it documents the level of consensus observed within each phase and across phases.

4. PROJECT DELIVERABLES BY PHASE

The deliverables of each project phase are described next, from the project kick off meeting in 1996 to the larger web-based survey in 1999.

1996 Kick Off Workshop: Process and Deliverables

Three days of discussion at the kick off workshop of the Software Engineering Standards Symposium resulted in (Jabir, 1997):

- Observations on the nature of fundamental principles;
- Criteria for identifying and evaluating candidate principles (see "What are fundamental principles?);
- Examples and counter-examples of principles;
- A recommendation that a Delphi study be organized and conducted among a group of software engineering experts using the criteria developed in Montreal and that a subsequent workshop be held to analyze the results of this Delphi study.

[5] See http://computer.org/tab/tclist/tcsoft.htm

First Delphi Study: Process and Deliverables

The objective of this first Delphi study was to identify an initial set of candidate fundamental principles. The desire to consult eminent representatives of the international community went hand in hand with the selection of the Delphi method. This technique consists of forming a panel of experts, who are not told each other's identities until the end of the exercise so that the prestige or power of one member of the group does not unduly influence the course of the discussion.

In Round 1, the international experts were asked to submit suggestions, based on the criteria already established in the kick off workshop. Each one was asked to draw up a list of the five fundamental principles they felt were most pertinent. This message was sent to 52 international software engineering experts. Thirteen individuals responded, which means that 65 potential principles were obtained.

Then, two Delphi study coordinators consolidated these 65 suggestions into a smaller number of principles that met the criteria of the SES'96 workshop. This produced a list of 16 potential fundamental principles. At this stage, the inclusion of the greatest number of suggestions possible was sought.

In Round 2, participants were asked to rate each of the 16 candidate fundamental principles from Round 1 on a scale of 1 to 10. They were also asked to comment on their ratings, which most of them did.

The goal of Round 3 of a Delphi study is to reinforce and confirm the ratings that emerge. Therefore, participants were sent the mean scores of each candidate fundamental principle, and asked whether or not they agreed with the rating and to add more comments if need be.

The value of this initial candidate list lies primarily in the expertise of the respondents. Table 2 lists the participants (12 of the 14[6]) who agreed to have their names published[7]. The other two participants chose to remain anonymous. We believe that the list represents a group that is diverse in terms of nationality, approach to software engineering, and theoretical versus practical experience.

Participant	Organization	Country
M. Azuma	Waseda University	Japan
F. P. Brooks	University of North Carolina	USA
R. N. Charette	ITHABI Corp.	USA
P. DeGrace	Consultant	USA
C. Ghezzi	Politecnico di Milano	Italy
T. Gilb	Result Planning Ltd.	Norway
B. Littlewood	City University	United Kingdom
S. MacDonell	University of Otago	New Zealand
T. Matsubara	Matsubara Consulting	Japan
J. Musa	Consultant	USA
R. S. Pressman	R.S. Pressman & Associates	USA
M. Shaw	Carnegie-Mellon University	USA

Table 2. First Delphi study – International experts

[6] One participant did not participate in round 1 of this study.

[7] For a short biography of each participant, consult the following address: http://www.lrgl.uqam.ca/fpse/emailfirstdelphi.pdf

It must be strongly emphasized that the output of a Delphi does not represent the single opinion of each individual participant, but rather a group view of the topics being investigated, and that it documents the degree of consensus (or lack of it) on such a group view.

ISESS'97 Workshop

This initial Delphi study was followed by a workshop at the IEEE-International Software Engineering Standards Symposium - ISESS'97[8], where some twenty participants discussed the findings. Based on this review and the lessons learned, this second workshop produced a list of improved criteria as well as a more refined list of fundamental principles, as illustrated in Table 5. These improvements were to be incorporated in the next Delphi study and in the web-based survey.

A.	Apply and use quantitative measurements in decision-making
B.	Build *with* and *for* reuse
C.	Control complexity with multiple perspectives and multiple levels of abstraction
D.	Define software artifacts rigorously
E.	Establish a software process that provides flexibility
F.	Implement a disciplined approach and improve it continuously
G.	Invest in the understanding of the problem
H.	Manage quality throughout the life cycle as formally as possible
I.	Minimize software component interaction
J.	Produce software in a stepwise fashion
K.	Set quality objectives for each deliverable product
L.	Since change is inherent to software, plan for it and manage it
M.	Since tradeoffs are inherent to software engineering, make them explicit and document them
N.	To improve design, study previous solutions to similar problems
O.	Uncertainty is unavoidable in software engineering. Identify and manage it

Table 3: List of candidate fundamental principles (in alphabetical order)

A few general conclusions from the initial Delphi group's responses can be drawn. Firstly, change management was a prime concern of the group and was rated as being more pertinent than discipline, measurement and control.

Secondly, the Delphi study participants were divided as to the type of measurements to consider and the type of control that should be exercised in software engineering: some favored a quantitative approach, but not all. However, the ISESS'97 workshop participants immediately stressed that the quantitative approach was inevitable in engineering, and that its importance could not be questioned if we are to speak of software "engineering."

[8] ISESS'97, Third International Symposium and Forum on Software Engineering Standards, Walnut Creek, CA, IEEE Computer Society, June 1997.

Finally, there was a concern about what constitutes the right amount of discipline, pursuit of quality and measurement. This goal is problematic, for it is very difficult to standardize an "appropriate" amount.

Second Delphi Study: Process and Deliverables

The first consultation, although very fruitful, had obvious limitations, notably the limited number of participants and the fact that their suggestions could have been consolidated differently, producing a different list of principles altogether. Consequently, a second Delphi study was carried out, this time among a group of software engineering officials from the IEEE Computer Society. The group consulted was therefore more representative of the users targeted by these candidate fundamental principles.

For the second Delphi study, 72 members of the Computer Society were contacted. Members contacted were on the *Technical Council on Software Engineering* or on editorial committees of IEEE-*Software* or IEEE-*Transactions on Software Engineering.* Thirty-one officials of the 72 contacted agreed to participate in this two round Delphi study.

Participants in the second Delphi study who agreed to have their names published are listed in Table 4. One participant chose to remain anonymous, one withdrew and one did not respond during Round 2.

Maarten Boasson	Richard Kemmerer	Shari Lawrence Pfleeger
Shawn Bohner	Barbara Kitchenham	Vaclav Rajlich
Terry Bollinger	Reino Kurki-Suonio	Rami R. Razouk
Andy Bytheway	David John Leciston	Sam Redwine
Carl Chang	Keith Marzullo	Mary Lou Soffa
James Cross II	Nancy Mead	David S. Wile
Peter Eirich	Stephen J. Mellor	Linda Wills
Bill Everett	Ware Myers	James Withey
Gene F. Hoffnagle	Michael Olsem	
Mehdi Jazayeri	Linda Ott	

Table 4: Delphi II - List of Participants

Web-based IEEE-CS SETC survey

The output of second Delphi study was to serve as an input to a web-based survey of the members of the Software Engineering Technical Council of the IEEE Computer Society, with the cooperation of the IEEE Computer Society.

A survey instrument was prepared and pre-tested with a limited sample of 50 names for whom the Computer Society had an email address, and of whom only 30 were valid at the time of the pre-test. Required adjustments to questions were made based on feedback received.

An introductory letter was prepared by the 1999 IEEE-CS president, Mr. Leonard Tripp, and sent by email to all members of the IEEE-SETC. A total of 3509 SETC members with valid email addresses were contacted. Out of this targeted audience, 565 members answered the web-based survey, representing approximately 16% of the targeted population.

Demographics of the respondents

- International participation: of the 556 respondents who indicated their country of residence, 50% were from the US, while the other 50% were from 48 countries, 11 countries having over 10 respondents each.

- Educational background: respondents had a significant mix of educational backgrounds ranging from computer science only, to engineering only, to math only, but mostly from any combination of these.
- Highest degree: 43% of the respondents indicated that they had a Ph.D, and 35% indicated a Master's degree.
- Years in software engineering: 36 % of the respondents indicated that they had over 20 years of experience in software engineering, while another 37% indicated that they have between 10 and 20 years of experience.
- Years of practice in industry: 23 % of the respondents indicated that they had over 20 years of experience in industry, and 36% between 10 and 20.
- Employer's line of business: the major categories of line of business of employers were: R&D (30%) and software (23%), and the balance from a variety of businesses, with only 6% from academia.
- Type of software: the larger portion of respondents by type of software were from the MIS domain with 38%, followed by real-time software (29%) and scientific software (14%).

Table 5 summarizes the degree of consensus on the candidate set of fundamental principles from each set of participants. For the first Delphi study with the group of 12 experts, the aggregate ratings of this group of experts is represented by the mean score (1 is low, and 10 is high), while in the second column the number of yes votes indicates the number of experts (out of 12[9]) who rallied to the mean score.

From second Delphi study on, it became apparent that the median was appropriate for this type of study, and the methodology was modified accordingly. The third column represents the median score, therefore, and the fourth column the number of yes votes out of 29[10] participants.

Finally, the fifth column indicates the median score for the 574 web-based participants, while the sixth column provides the standard deviation for this larger sample.

	Delphi 1		**Delphi 2**		**Survey**	
	Mean Score	Yes Votes	Median Score	Yes Votes	Median Score	Std Dev
A. Apply and use quantitative measurements in decision-making	7.6	7/12	7	13/29	8	2,4
B. Build with and for reuse	8	7/12	9	17/29	8	2,3
C. Control complexity with multiple perspectives and multiple levels of abstraction	N/A	N/A	8	23/29	8	2,4
D. Define software artifacts rigorously	6.4	6/12	8	22/29	8	2,5
E. Establish a software process that provides flexibility	7.6	7/12	8	21/29	8	2,3
F. Implement a disciplined approach and improve it continuously	6.9	4/12	8	19/29	9	2,4
G. Invest in understanding the problem	8.7	7/12	10	29/29	10	2
H. Manage quality throughout the life cycle as formally as	7.8	7/12	9	20/29	8	2,5

[9] 12 out the 14 first Delphi study participants took part in round 3.
[10] 29 out the 31 second Delphi study participants took part in round 2.

possible						
I. Minimize software component interaction	7.3	8/12	9	25/29	7	2,7
J. Produce software in a stepwise fashion	7.7	7/12	8	23/29	7	2,7
K. Set quality objectives for each deliverable product	7.7	8/12	8	20/29	8	2,3
L. Since change is inherent to software, plan for it and manage it	9.1	9/12	10	26/29	9	2
M. Since tradeoffs are inherent to software engineering, make them explicit and document them	8.4	8/12	9	25/29	9	2,3
N. To improve design, study previous solutions to similar problems	N/A	N/A	9	24/29	8	2,1
O. Uncertainty is unavoidable in software engineering. Identify and manage it	8	8/12	10	25/29	8	2,5

Table 5: Overview of participant consensus - documented at project phases

5. SUMMARY AND NEXT STEPS

This paper has reported on a series of efforts undertaken to try and identify a set of fundamental principles of software engineering. A first workshop was held at the Forum on Software Engineering Standards Issues of 1996 (SES'96) to establish what a fundamental principle is and which criteria it should conform to.

A Delphi study was then conducted in 1997 over the Internet among 14 international software engineering experts to identify a first candidate list of fundamental principles of software engineering. A second workshop was held at the International Symposium on Software Engineering Standards of 1997 (ISESS'97) to eliminate or reformulate some of the principles and the criteria. Subsequently, a second Delphi study was conducted in 1998 among 31 IEEE software engineering officials in order to improve the principles. From these studies, a list of fifteen candidate fundamental principles of software engineering was compiled. Finally, an electronic survey was conducted among the membership of the Software Engineering Technical Council to verify the relevance of these candidate principles for practitioners and to help determine which of these fifteen candidate principles are indeed fundamental.

The current set of fundamental principles was developed based on domain experts' opinions, in successively larger samples. While highly valuable in proceeding quickly and at a low cost to develop and document the level of consensus on the process output, this type of research has inherent methodological limitations which should be addressed in subsequent years.

Further investigations are therefore needed, but techniques other than opinion surveys should now be investigated, through empirical designs for verifying these principles both in current theories proposed in the field of software engineering and by observation of their implementation in currently recommended best practices.

Other activities can be planned as well: analysis of the body of current standards, of generally accepted knowledge in software engineering as described in the Guide to the Software Engineering Body of Knowledge, and of university programs in software engineering to assess the extent to which fundamental principles are covered.

In addition to the quantitative aspects of the survey, a significant number of very valuable comments were provided by the participants. A thorough and structured analysis of these comments would provide very valuable research material for further exploration. An example of such an analysis on the role of measurement in the set of candidate fundamental principles can be found in (Wolff, 1999).

Through a judicious combination of these proposed next steps, it is hoped that potential flaws of the opinion-based studies presented in this paper will be pinpointed and that the set of candidate principles can be judged on the basis usability, relevance, significance and usefulness.

Acknowledgments

We would like to thank everyone who contributed their ideas to this work: participants of Delphi studies, workshops and web-based survey. The Software Engineering Management Research Laboratory at the Université du Québec à Montréal is supported through a partnership with Bell Canada. Additional funding is provided by the Natural Sciences and Engineering Research Council of Canada.

The authors would also like to thank the IEEE Computer Society and especially John Keaton for their continuing support.

References

Boehm, B. W., Seven Basic Principles of Software Engineering, *The Journal of Systems and Software*, Vol. 3, no 1, March, 3-24 (1983).

Bourque, P., Dupuis, R., Abran, A., Moore, J.W., Tripp, L., The Guide to the Software Engineering Body of Knowledge, *IEEE Software*, Vol. 16, no 6, November /December, 35-44 (1999).

Davis, A.M., *201 Principles of Software Development*, McGraw-Hill, New York, 1995, p. 240.

IEEE Std 610.12-1990, 1991, *IEEE Standard Glossary of Software Engineering Terminology*, Corrected Edition, February.

Jabir, Moore, J.W., A Search For Fundamental Principles of Software Engineering, *Report of a Workshop Conducted at the Forum on Software Engineering Standards Issues* (1996), published in *Computer Standards & Interfaces - The International Journal on the Development and Application of Standards for Computers, Data Communications and Interfaces*, Vol. 19, no 2, 155-160 (1998), North-Holland, Elsevier Science. (The participants at this workshop dubbed their group, "Jabir.") Also available at http://www.lrgl.uqam.ca/publications/pdf/249.pdf

McConnell, S., Software's Ten Essentials, *IEEE Software*, March/April, Vol. 14, no 2, 143-144 (1997).

McConnell, S., Software Engineering Principles, *IEEE Software*, March/April, Vol. 16, no 2, 6-8 (1999).

Moore, J.W., *Software Engineering Standards – A User's Road Map*, IEEE Computer Society Press, Los Alamitos, California, 1997, p. 328.

The results of the 1996 Software Engineering Standards conference can be found at URL: "http://www.lrgl.uqam.ca/ses96.html".

Tripp, L. and P. Voldner, A Market-Driven Architecture For Software Engineering Standards, *Proceedings of the Second IEEE International Software Engineering Standards Symposium (ISESS'95)*, IEEE Computer Society, 105-116 (1995).

Wolff, S., La Place de la Mesure au Sein des Principes Fondamentaux du Génie Logiciel, Masters Thesis, Université du Québec à Montréal, 1999.